Energy-Aware Video Streaming on Smartphones

Wenjie Hu and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
E-mail: {wwh5068, gcao}@cse.psu.edu

***Abstract*—Video streaming on smartphone consumes lots of energy. One common solution is to download and buffer future video data for playback so that the wireless interface can be turned off most of time and then save energy. However, this may waste energy and bandwidth if the user skips or quits before the end of the video. Using a small buffer can reduce the bandwidth wastage, but may consume more energy and introduce rebuffering delay. In this paper, we analyze the power consumption during video streaming considering user skip and early quit scenarios. We first propose an offline method to compute the minimum power consumption, and then introduce an online solution to save energy based on whether the user tends to watch video for a long time or tends to skip. We have implemented the online solution on Android based smartphones. Experimental results and trace-driven simulation results show that that our method can save energy while achieving a better tradeoff between delay and bandwidth compared to existing methods.**

I. Introduction

Video streaming has seen tremendous growth in recent years on mobile devices such as smartphones and tablets. Mobile video streaming on Youtube, Vimeo, Netflix, has increased 70% per year recently, and will take about 70% of the total traffic by 2018 [4]. Since videos have much larger data size when compared to emails and webpages, lots of energy will be consumed to download these videos on smartphones. Since advances in battery technology have been slow, we have to design network protocols to improve the energy efficiency of video streaming on smartphones.

On smartphones, the power consumption of the wireless interface is high when it is turned on during data transmission, and is much lower when the interface is turned off. In 3G/4G LTE networks, the wireless interface will stay on an intermediate high power state (called *tail state*) for some time (as long as 15 seconds) after a data transmission. To save the energy, Zhao *et al.* [19] proposed to aggregate data traffic when browsing webpages. Thus a simple solution to save energy during video streaming, called ON-OFF scheme [17], is to download as much video as possible, and then turn the wireless interface off when playing the video. When the video buffer is almost empty, the wireless interface is turned on again to download more data.

The ON-OFF scheme may not work well. This is because, most of time, a user may skip frequently when watching a video and may quit long before the end of the video. Then, downloading too much video could waste lots of energy and bandwidth. To reduce the wastage, bitrate streaming [7] can be used, which downloads video at the video encoding rate (video playback speed), i.e., downloading video and using it immediately. From another point of view, bitrate streaming is a scheme which sets the video buffer to be extremely small. Bitrate streaming can reduce the bandwidth wastage but it keeps the wireless interface on all the time, and thus costs much more energy. Moreover, bitrate streaming may not download the content in time when the video content size varies or the network throughput fluctuates, and then introducing rebuffering delay which affects users' Quality of Experience (QoE) [13].

Energy efficiency during video streaming has attracted lots of research recently. The work in [7] summarized various streaming techniques used by commercial video providers, such as ON-OFF, bitrate streaming, throttling, and etc, and compared them in terms of power, bandwidth, and delay. In [6], the authors measured the energy of video streaming when using WiFi and 3G. However, they did not provide any effective solution to improve the energy efficiency. To save energy, Li *et al.* [14] proposed GreenTube to dynamically select the buffer size based on the remaining time a user will watch the video, so as to avoid downloading too much data when users skip or quit, but it relied on an accurate prediction of user watching behaviors. Horque *et al.* [8] use crowd-sourced trace to predict when a user quits a video, so as to reduce the bandwidth wastage and thus saving energy, but the prediction may not be accurate for individual users. Moreover, none of them considers the bandwidth and energy wastage when users skip to watch video, which is common in real life.

In this paper, we design energy-aware video streaming solutions to satisfy the delay requirement and reduce the bandwidth wastage by considering user skip and early quit scenarios. We first propose an offline method to compute the minimum energy to download a video, while satisfying the delay requirement. This offline solution requires knowledge of future video watching information, thus, we propose an online energy-aware video streaming scheme to save energy and bandwidth. We classify video streaming into two modes: *stable* mode and *unstable* mode, based on whether the user tends to watch video for a long time or tends to skip. In the stable mode, an optimized ON-OFF scheme is used to save energy and bandwidth. In the unstable mode, a modified bitrate streaming scheme is used to minimize the downloading

This work was supported in part by the National Science Foundation (NSF) under grant number CNS-1218597 and CNS-1421578.

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

of wasted content while not increasing the rebuffering delay.

To verify the effectiveness of the proposed solution, we have implemented the online scheme on Android based smartphones. Both experimental results and trace-driven simulation results show that our scheme can reduce the energy consumption. To summarize, our contributions are as follows:

- We analyze the power consumption during video streaming considering user skip and early quit scenarios, and propose an offline solution to find the minimum energy while satisfying the delay requirement.
- We propose an online energy-aware video streaming scheme to reduce the energy consumption by applying different schemes based on whether a user tends to watch a video for a long time or tends to skip or quit.
- A prototype has been developed on Android platform and the results show that our solution can save energy while achieving a better tradeoff between delay and bandwidth compared to existing methods.

The rest of this paper is organized as follows. Section II discusses related work. Section III introduces the background and motivation. In Section IV, we define and analyze the min-energy video streaming problem. Section V presents the online energy-aware solution and Section VI presents its implementation. Section VII evaluates the performance of our solution and Section VIII concludes the paper.

II. Related Work

The wireless interface consumes most of the energy on mobile devices [16]. The common approach to save energy is to aggregate the data traffic and then put the wireless interface to sleep most of time. For example, Hu *et al.* proposed a protocol to save energy by aggregating the traffic of multiple users and then sending the data to/from the selected proxy node using energy efficient P2P interfaces [9], [10]. Similar ideas on video streaming have been proposed in [15], [12]. However, they either require one user to download large video for others in the group [15] or require users to watch the same video at the same time [12]. Traffic aggregation on a single node can save energy [19], but downloading the whole video will increase the delay and require much more buffer space.

Video streaming has special characteristics when compared to other data traffic. First, users may frequently skip during video watching, and thus buffering large video may waste bandwidth and energy. Second, video streaming is time sensitive. Video is organized as Group of Pictures (GoPs) with strict time constraint [18], and each GoP should be downloaded before being played; otherwise it will degrade QoE [13], [11].

Energy-aware video streaming should consider delay and bandwidth consumption. GreenBag [5] uses both LTE and WiFi interfaces to download video, which can reduce delay, but using two interfaces may increase the energy and bandwidth consumption. To achieve a balance, GreenTube [14] dynamically selects the buffer size from a given set based on when the user will watch the video, so as to save energy and bandwidth by selecting a smaller buffer when users skip or quit. However, the prediction is based on the watching history of other videos, which may not be accurate for individual users. Hoque *et al.* use crowd-sourced video to determine the position that a user will quit the video [8]. They considered the early quit problem but did not consider user skipping within a video. Different form these works, our energy-aware video streaming considers both user skip and early quit scenarios, and leverages user actions to improve the prediction accuracy.

III. Background and Motivation

A. Basics of Video Streaming

A video is organized as a number of frames. Generally, a movie contains 24 frames per second. As the frame size is pretty large, it is compressed before streaming out, and then decompressed at the destination. This process is referred to as encoding and decoding. In the commonly used video coding standard H.264/AVC/MPEG-4 Part 10, the video frames are classified into I, P, B frames, where P and B frames have much smaller size than the I frames, but the decoding of P (B) frames depends on the previous (and later) I/P frames. Inserting more B frames reduces the data transmission size but increases the decoding time at the client side. To achieve a balance, a given number of different frames are organized in a proper sequence, which is called a Group of Picture (GoP). For example, one GoP may be organized as IBBBPBBBPBBBPBBB.

Since each GoP contains a fixed number of frames, and each frame has the same length, e.g., $\frac{1}{24}$ second, the playback time of a GoP is fixed. However, the size of GoP varies and thus the time to download a GoP varies. Fig. 1 shows the GoP size of one movie. The minimum size of the GoP is 363 Bytes, and the maximum is 256 KB, which is 700 times larger than the minimum. Moreover, the GoP size has no particular pattern and can not be predicted by neighbor GoPs.

B. Energy Consumption during Video Streaming

In this paper, we focus on video streaming over 4G LTE networks, and the same idea can be applied to other wireless networks such as 3G. In LTE, the wireless interface is controlled by the RRC (Radio Resource Control), and can work in three states. When there is no data transmission, the phone stays at IDLE state in which it consumes very little power. When a data transmission request comes, the phone promotes to the RRC_CONNECTED state and begins to transfer data. The promotion delay is t_{pro}. When the data transmission finishes, the phone switches to DRX (Discontinuous Reception) state, which is also called tail state. The tail length is controlled by a timer t_{tail}. When the timer expires, the phone switches to the IDLE state again. Among these states, the data transmission state consumes the highest power, while the tail state consumes less and the IDLE state consumes very little power. The power consumption of promotion, data transmission, and tail (DRX) states are denoted as P_{pro}, P_{data} and P_{tail}, respectively. The data throughput of LTE is denoted as r.

The content of a video is divided into GoPs, where the ith GoP g_i has a data size d_i and a playback time (length) l_i, and is downloaded at time t_i (t_1 =0 and $i \geq 1$). The

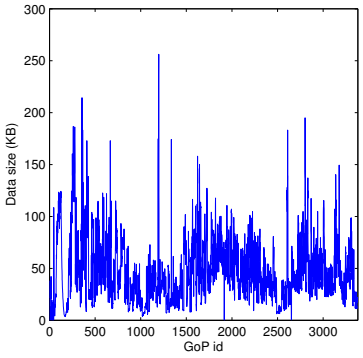


Fig. 1. The variance of GoP size

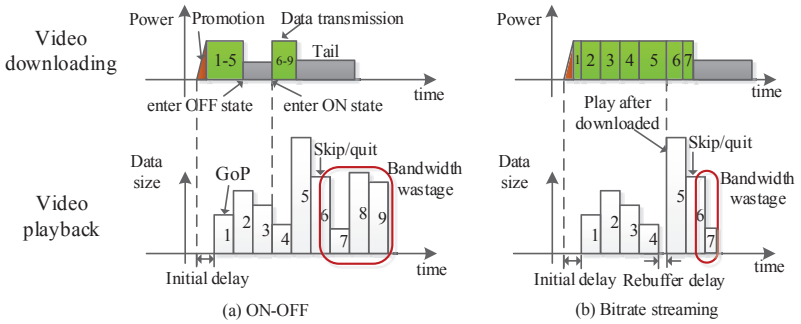


Fig. 2. The power consumption and delay of two video streaming schemes

power consumption and the delay of g_i are related to the previously downloaded GoP g_{i-1}. The time interval between g_i and g_{i-1} is denoted as $\Delta t = t_i - t_{i-1} - d_{i-1}/r$. As GoPs have strict sequence, g_i should always be downloaded after g_{i-1}, and thus $\Delta t \geq 0$. The power consumption of g_i can be computed in two cases: 1) if $\Delta t > t_{tail}$, g_i is downloaded when the LTE interface is already demoted to IDLE state, so it needs to pay extra promotion and tail energy, besides the data transmission energy; 2) Otherwise, g_i only consumes part of the tail energy. In summary, the energy of g_i can be computed by Eq. 1. The time used to download g_i is denoted as T_i and can be computed similarly according to the two cases, which is shown in Eq. 2. If $\Delta t > t_{tail}$, we should consider the promotion delay, otherwise we only need to consider the data transmission delay.

$$E_i = \begin{cases} P_{pro} \times t_{pro} + P_{data} \times \frac{d_i}{r} + P_{tail} \times t_{tail}, & \text{if } \Delta t > t_{tail} \\ P_{data} \times \frac{d_i}{r} + P_{tail} \times \Delta t, & \text{Otherwise} \end{cases} \quad (1)$$

$$T_i = \begin{cases} d_i/r + t_{pro}, & \text{if } \Delta t > t_{tail} \\ d_i/r, & \text{Otherwise} \end{cases} \quad (2)$$

We use Samsung Galaxy S5 with AT&T LTE data plan for evaluation. Monsoon power monitor is used to provide power supply to the smartphone and then sample the power value at a rate of 5000 Hz. We use *iperf* to measure the throughput. We also root the smartphone and capture the network traces using wireshark. From the network trace and the power trace, we can differentiate the states of LTE and measure the power of each state. The measured power and throughput values are shown in Table I, where each result is averaged over 10 tests. The power value is the whole phone's power consumption when the screen is on.

C. Motivations

Mobile video providers such as Youtube, Vimeo and Netflix, adopt different techniques to stream videos [7]. Generally speaking, the existing video streaming techniques can be

TABLE I
POWER AND THROUGHPUT IN DIFFERENT STATES OF LTE

State	Power (mW)	Duration (s)	Throuput (Mbps)
Promotion	1548.58 ± 47.10	0.63 ± 0.36	-
Data	1568.26±54.95	-	19.01±2.59
Tail	1266.62±35.76	10.27±1.97	-

divided into two categories, as shown in Fig. 2. The upper figures show the power consumption of the smartphones when downloading these GoPs. The number inside the data transmission state shows the GoPs that are downloaded during this time. The lower figures show the video playback, i.e., the GoP size and their playback time. Each bar indicates one GoP and the number inside is the GoP id. The two types of figures are synchronized, where the dotted line indicates the same time.

The first category is *ON-OFF* streaming, as shown in Fig. 2(a). It sets a buffer size for a video and downloads the video at the highest speed until the buffer is full. Then it turns off the wireless interface. For example, the phone downloads five GoPs and turn the wireless interface off. When the buffered content is less than a threshold, e.g., less than 40% of the buffer time in Youtube [8], the smartphone turns on the wireless interface again to download data until the buffer is full again. In Fig. 2(a), The wireless interface is turned on again after playing GoP 3 and then GoP 6-9 are downloaded. One extreme case is to download the whole video and then play it.

The second category is *Bitrate Streaming*, as shown in Fig. 2(b). This scheme downloads content at the encoding rate (playback speed) of the video. It downloads a GoP when it is about to play this GoP. As the video playback lasts for a long time, this scheme keeps the wireless interface on all the time.

Comparisons of the two schemes:

The ON-OFF scheme can save energy by turning the wireless interface off, but it wastes a large amount of bandwidth and energy when users skip. For example, in Fig. 2(a), if the user skips when playing GoP 6, the energy and bandwidth used to download GoP 7-9 and part of 6 are wasted. On the other hand, bitrate streaming wastes less bandwidth. As

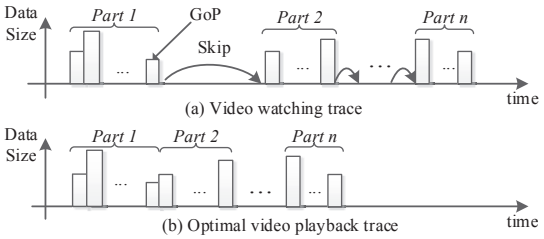


Fig. 3. The watching trace of a typical video: (a) users may skip to watch several parts of the video, (b) a user expects all video parts should be played back without delay, even when he skips.

shown in Fig. 2(b), when a user skips at the same position, the wasted bandwidth is much less than that of the ON-OFF scheme. However, bitrate streaming may introduce rebuffering delay, which means a GoP is not downloaded before its playback time, such as GoP 5 in Fig. 2(b). The rebuffering delay degrades QoE and should be avoided. Both schemes have advantages and disadvantages. ON-OFF scheme works better when the user watches a video for a long time, and bitrate streaming works better when the user tends to skip or quit the video. However, recent video streaming providers only adopt one particular scheme, without considering other scenarios. In this paper, we propose a solution which considers user watching behaviors and then adaptively apply the proper scheme to save energy while reducing the bandwidth wastage and the rebuffering delay.

IV. Min-Energy Video Streaming

In this section, we formalize the min-energy video streaming problem, considering that a user may skip or quit before the end of the video.

A. Problem Statement

A video is divided into a number of GoPs, and the GoP is used as the minimum downloading unit. GoP g_i has a playback time (length) of l_i and a data size of d_i. A general video watching trace is shown in Fig. 3(a). During video streaming, a user watches part of the video, and then skips forward to watch another part, and so on. This pattern repeats for n times, and the user watches N GoPs in total. In such scenario, the expected (optimal) video playback trace should work like Fig. 3(b), where all GoPs should be displayed smoothly without delay even when the user skips. Assuming the user watching trace is known, we find the optimal video playback solution to minimize the energy consumption.

The energy to download g_i can be computed using Eq. 1. It is related with the time to download, i.e., t_i, and the data size d_i. The time t_i should satisfy two conditions. First, all GoPs should be downloaded in sequence, i.e., g_i should be downloaded after g_{i-1}. Considering the downloading time, we have $t_i \geq t_{i-1}+d_{i-1}/r$. Second, to satisfy the delay requirement, g_i should be downloaded before it is played, i.e., before the end of playback of g_{i-1}, which means $t_i \leq \sum_{j=1}^{i-1} l_j - T_i$, where T_i is the time used to download g_i and can be computed by Eq. 2.

A buffer with maximum size B is allocated to video streaming. Before g_i is downloaded, the video size that has already been downloaded is $\sum_{j=1}^{i-1} d_j$. At the same time, the video content that has been consumed (played) is $\sum_{j=1}^{p} d_p$, where p is the maximum id of the GoP that has been displayed and is computed by Eq. 3. A GoP g_i is downloaded only when there is enough buffer for it, i.e., when $\sum_{j=1}^{i-1} d_j - \sum_{j=1}^{p} d_p + d_i \leq B$.

$$p = \arg\max \sum_{j=1}^{p} l_j < t_i \tag{3}$$

Putting them together, our goal is to minimize the total energy as shown in Eq. 4 (E_i was defined in Eq. 1). The problem is summarized as follows, where t_1 initialized to 0 and $i > 1$ in Eq. 5 to Eq. 7.

$$\textbf{minimize} \ \sum_{i=1}^{N} E_i \tag{4}$$

$$\textbf{subject to:} 0 \leq t_i \leq \sum_{i=1}^{N} l_i$$

$$t_i \geq t_{i-1} + d_{i-1}/r \tag{5}$$

$$t_i \leq \sum_{j=1}^{i-1} l_j - T_i \tag{6}$$

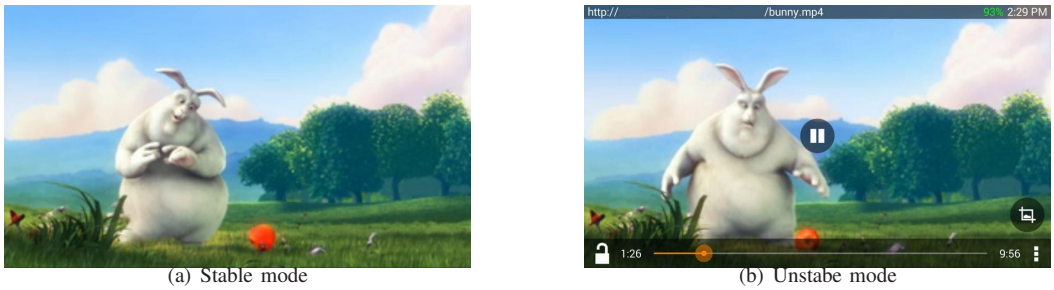
$$\sum_{j=1}^{i-1} d_j - \sum_{j=1}^{p} d_p + d_i \leq B \tag{7}$$

B. Our Solution

By solving Eq. 4, we obtain the minimum energy. However, the computation consumes lots of time. Thus we use another solution. Based on Eq. 1, the energy of video streaming using LTE includes promotion, data transmission, and tail energy. Considering that the data transmission energy is determined by the video size which is fixed, we can only reduce the promotion and tail energy. As there is no bandwidth wastage, we use the ON-OFF scheme to minimize energy by: 1) aggregating as much traffic as possible in the ON state; 2) leaving the LTE interface in the OFF state as long as possible.

ON state: In the ON state, we first download video and fill the buffer. As the content is also consumed during downloading, there will be extra space after downloading B size of data. If it is enough for more GoPs, these GoPs are downloaded and there is no extra tail energy as the LTE interface is still in the data transmission state. This process repeats until the buffer is full. Suppose the last download GoP is g_m. After switching to the OFF state, and the LTE interface enters tail state. After t_{tail} time, it demotes to IDLE state.

OFF state: The OFF state should last as long as possible. Considering the delay requirement, the next GoP g_{m+1} should be downloaded just before the end of playback of g_m, i.e., $t_{m+1} = \sum_{i=1}^{m} l_i - T_{m+1}$. So we enter the ON state at t_{m+1} and begin to download data. The downloading of the last part of video should be considered separately. When there only remains $b < B$ of data after the ON state, we do not wait



(a) Stable mode (b) Unstabe mode

Fig. 4. Two modes during video streaming. In stable mode (a), the progress bar is not shown, which indicates the user tends to watch video for a long time. In unstable mode (b), the user triggers the progress bar and tends to skip to other places.

until t_{m+1}, but begin to download these data when there is enough buffer for it. The reason is that, if the downloading of b happens when the LTE interface is still in the tail state, the promotion energy and part of the tail energy can be saved. Since the LTE downloading speed is faster than the video encoding speed, GoPs after g_{m+1} can also be downloaded within the delay constraint when g_{m+1} is played.

V. Online Energy-Aware Video Streaming

In the previous section, the min-energy video streaming assumes that the user watching behavior is known. Since this may not be possible in practice, we propose an online energy-aware video streaming scheme in this section.

A. Problem Statement

The user watching behavior is hard to predict in practice since it is related to user interest, video category, time and location, user's emotion, remaining energy, and etc. Thus, we use a different approach; i.e., predicting user's future behavior based on his current actions.

We classify the mobile video streaming into two modes: *stable* mode and *unstable* mode. An example of the two modes is shown in Fig. 4. In the stable mode (Fig. 4(a)), the progress bar is not shown, which indicates that the user tends to watch the video for a relatively long time. In the unstable mode (Fig. 4(b)), the user triggers the progress bar by touching the screen, which indicates that he tends to skip to other parts. Different solutions are used in each mode to save energy, delay and bandwidth.

B. Video Streaming in Stable Mode

In stable mode, the user is likely to watch video for a relative long time. Thus, we use a modified ON-OFF scheme to save energy. As a large amount of energy is wasted due to downloading unused content in the original ON-OFF scheme, our modification focuses on saving energy by reducing bandwidth wastage. There are two buffer thresholds in our method: an upper buffer threshold (α) and a lower buffer threshold (β). When the downloaded data is larger than α, the wireless interface is turned off. When the buffered data is smaller than β, the wireless interface is turned on again to download video. The key challenge is to select the proper buffer thresholds.

The upper buffer threshold α is determined as follows. When there is no progress bar, α is set to the maximum buffer size B. When a user touches the screen and triggers the progress bar, it is reduced to the lower buffer threshold. This also switches the video streaming to unstable mode.

The selection of β is more complex, especially when considering the bandwidth wastage due to user skipping. Suppose a user has downloaded data and then skips at position s in the buffer. In the worst case, all downloaded data are wasted. The bandwidth wastage is denoted as $W(s)$, and there are three cases to compute $W(s)$, as shown in Fig. 5.

- When s is smaller than $B-\beta$, as shown in Fig. 5(a), the phone is still in the OFF state, so the bandwidth wastage is $B-s$.
- When s is bigger than $B-\beta$, the phone enters ON state. If the new downloaded content has not filled the whole buffer, i.e., when $s < \frac{r}{r-v}(B-\beta)$, where r is the network throughput and v is the average video playback speed, as shown in Fig. 5(b), the new downloaded data is $\frac{\beta-B-s}{v} \times r$. The total bandwidth wastage is $B-s+\frac{\beta-B-s}{v} \times r$.
- If the phone has entered into ON state for a longer time, the whole buffer will be filled again, as shown in Fig. 5(c). Then, the bandwidth wastage is B.

Since a user may skip at any place in the video, we assume the probability is uniformly distributed within the buffer, i.e., $p_s = 1/B$. Based on this, the expected bandwidth wastage can be computed by Eq. 8:

$$\begin{aligned} E(w) &= \int_0^B W(s) p_s \, \mathrm{d}s \\ &= \frac{1}{B}\Bigg(\int_0^{B-\beta} (B-s)\mathrm{d}s \\ &\quad + \int_{B-\beta}^{\frac{r}{r-v}(B-\beta)} \Big((1-\frac{r}{v})(B-s) + \frac{r}{v}\beta\Big)\mathrm{d}s \\ &\quad + \int_{\frac{r}{r-v}(B-\beta)}^{B} B \, \mathrm{d}s \Bigg) \end{aligned} \tag{8}$$

We aim to select a proper β to minimize $E(w)$. By letting $\frac{\mathrm{d}E(w)}{\mathrm{d}\beta} = 0$ we get the optimal β that minimizes the bandwidth wastage in Eq. 9.

$$\beta_{minwaste} = \frac{r}{2v-r} B \tag{9}$$

From the delay point of view, β should be big enough so there is enough time to download at least one GoP. As the size of next GoP, say g_n, is unknown before downloading,

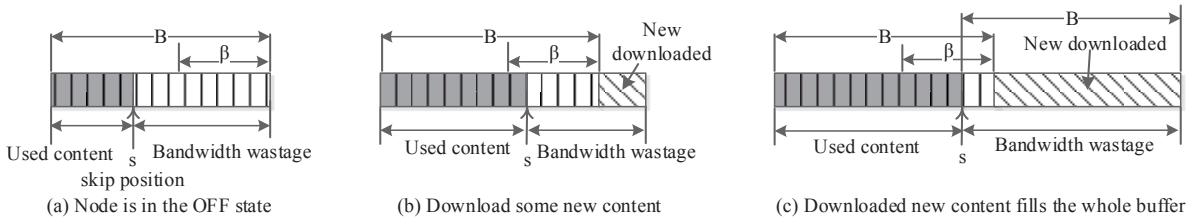


Fig. 5. The bandwidth wastage when a user skips at place s when using ON-OFF streaming. In (a), the phone is in the OFF state. In (b) and (c), the phone enters ON state and downloads new content.

we assume it has the largest size among those downloaded GoPs and the wireless interface is in the OFF state when downloading. When the bufferred data size is less than the value computed in Eq. 10, we begin to download g_n.

$$\beta_{mindelay} = (\frac{\max d_j}{r} + t_{pro}) \times v, 1 \leq j \leq n-1 \quad (10)$$

The lower buffer threshold should consider both bandwidth wastage and delay, and then it should be the maximum of these two, as shown in Eq. 11.

$$\beta = \max\{\beta_{minwaste}, \beta_{mindelay}\} \quad (11)$$

C. Video Streaming in Unstable Mode

In the unstable mode, users tend to skip to another part of the video, and then buffering more data may waste bandwidth. Thus, we use the bitrate streaming scheme in unstable mode. Bitrate streaming downloads g_i when it begins to display g_{i-1}. When the GoP size or network condition fluctuates, future GoPs may not be downloaded in time, and thus increases the rebuffering delay and affects the QoE. To address this problem, enough time is reserved to download g_i considering the network condition, and the buffer size is set to $\beta_{mindelay}$ as in Eq. 10.

After a user drags the progress bar, he may begin to view another part of the video for a long time. In this case, although the progress bar is still visible, we gradually increase the buffer size by doubling it after the playback of each GoP, until it reaches B. When the progress bar disappears and the buffer size is smaller than B, it increases to B immediately and enters stable mode.

VI. Implementation

The online energy-aware streaming has been implemented on Android based phones. We use Samsung Galaxy S5 running Android 4.4 as the testbed. The video application is based on VLC 2.1.3 [3]. VLC is a platform that supports coding/decoding in various formats, such as H.264/AVC, DivX and MKV, and supports streaming using different protocols such as HTTP, RTSP and UDP. One advantage of VLC is that it is highly "modularized" on interface, audio output, video output, stream output, and etc. Thus, it is easy to modify or add one module without interrupting the system.

In our deployment, we mainly modify the `stream_output` and `user_interfaces` module. The default video streaming scheme of VLC is bitrate streaming, which keeps the wireless interface on all the time. We also added ON-OFF streaming and our online energy-aware streaming in the `stream_output` module. For our scheme, we modify the user interface and insert a listener for the progress bar. We monitor the appearance and disappearance of the progress bar and also check whether it is dragged. This information is used to trigger different video streaming modes. The timestamp of each activity is also recorded. For the ON-OFF streaming and our streaming scheme, the maximum buffer size is set to 10 MB by default.

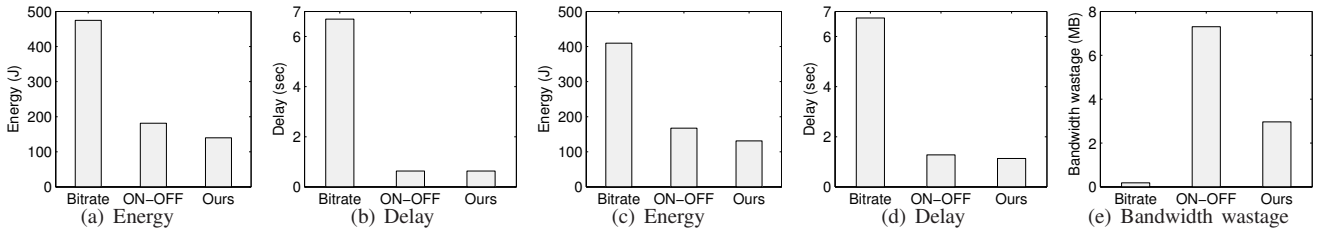
VII. Evaluations

In this section we compare the performance of our proposed approaches with existing approaches using testbed and trace-driven simulations.

A. Evaluation Setting

On the testbed, we evaluate the performance of our online energy-aware video streaming with bitrate streaming and ON-OFF streaming. In the simulation, we also consider the min-energy video streaming method and GreenTube [14]. A brief summary of these methods are listed as follows:

- *Bitrate streaming*: This method downloads video at the video encoding speed (playback speed), i.e., it downloads GoP $i+1$ when it begins to play GoP i.
- *ON-OFF*: ON-OFF uses a constant buffer size, which is 10 MB in this paper. Suppose the data in the buffer can be played for t time. The wireless interface is turned on when the buffered data is less than $t * 40\%$, and it is turned off when the buffer is full.
- *GreenTube*: GeenTube adaptively adjusts the maximum buffer size every second based on the prediction of the remaining time that a user will watch the video before skipping or quit. The maximum buffer size is selected from a set and the one that can minimize the power consumption is used. Then it performs similar as the ON-OFF scheme. The candidate buffer size set is $\{2, 5, 8, 10\}$ MB in this paper. Suppose the accurate remaining time before a user skips is t_r, we inform GreenTube a random value ranges from $0.8t_r$ to $1.2t_r$.



(a) Energy (b) Delay (c) Energy (d) Delay (e) Bandwidth wastage

Fig. 6. The energy, delay and bandwidth wastage of different methods on testbed. Here (a) and (b) are the results when users watch the whole video, and (c)-(e) are the results when users skip twice.

TABLE II
VIDEO TRACES IN SIMULATION. THE BOLD WORD IN A VIDEO NAME IS USED TO INDICATE THAT VIDEO IN THE SIMULATION.

Video Name	Length (min)	Total Size (MB)	GoP Number	Min GoP Size (KB)	Max GoP Size (KB)
Sony Demo	9.82	133.65	1105	1.16	340.84
NBC News	27.51	583.28	3095	45.41	476.46
Silence of the **Lambs**	30	151.74	3374	0.31	429.53
Star **Wars** IV	30	153.27	3374	0.35	256.27
Tokyo Olympics	73.96	859.94	8320	0.34	402.02

- *Ours:* This is our online energy-aware video streaming method in Section V. The maximum buffer size is set to 10 MB. We assume a user begins to skip after $1-3$ seconds when the progress bar is trigged out, which is obtained from the testbed.
- *Optimal:* This is the min-energy video streaming using the offline method in Section IV. It computes the minimum energy needed to download the video.

When a user skips to watch a video, we assume the skip duration (e.g., the length a user drags the progress bar) is uniformly distributed from 40 to 100 seconds. If one skip goes over the length of a video, we assume the user quits the video before ending. When there are skips, all methods skip at the same time. Each result in the skip scenario is averaged over five tests.

For the performance metrics, our focus is the energy consumed during video streaming. We also consider the delay and bandwidth wastage. The delay contains both initial delay and rebufferring delay, as shown in Fig. 2.

B. Testbed Based Evaluations

In the testbed, we use the modified VLC media player to watch videos and use the Monsoon power monitor to measure the power consumption directly. We use the 480p "Big Buck Bunny" as the video trace [1], which has a size of 64.7 MB, and lasts for 10 minutes. This video is first downloaded to a local server in our lab to avoid Internet delay variation. The WiFi interface is turned off and all downloading is via the LTE interface.

We watch the video using bitrate streaming, ON-OFF, and our online method in two cases. First, we watch the whole video without skipping. Second, we skip twice randomly. Each time we skip for about 1 minute. For the three methods, we try our best to skip at the same time and with the same duration.

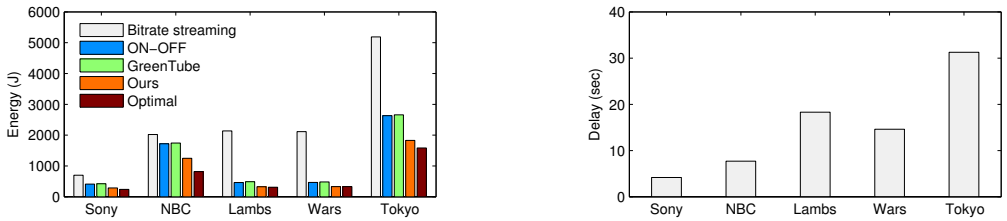
1) Video Streaming without Skipping: The energy and delay when users watch the whole video are shown in Fig. 6(a) and Fig. 6(b), respectively. As the user watches the whole video, our method works in stable mode all the time. Generally speaking, Bitrate streaming consumes much more energy and introduces longer delay than the ON-OFF method and our method. Since the user watches the whole video, no bandwidth is wasted, and then the advantage of bitrate streaming is not noticeable.

In Fig. 6(a), our method and ON-OFF save 70.5% and 61.8% of energy than bitrate streaming, since both periodically turn the wireless interface off to save energy. Our method further saves 23.1% of energy than ON-OFF, since the lower buffer threshold in our method is much smaller than that of ON-OFF, and thus the wireless interface stays in OFF state longer. From the delay point of view, both our method and ON-OFF only pay for the initial delay when downloading the first GoP, which is really small. However, bitrate streaming pauses several times when the next GoP is larger and thus introduces 10 times more delay than the other two methods.

2) Video Streaming when Users Skip: We also compare the performance of the three methods when users skip twice within the video and the results are shown in Fig. 6(c) to 6(e). Bitrate streaming has the smallest bandwidth wastage but consumes much more energy and has much longer delay. On the other hand, ON-OFF has less power consumption and delay, but consumes much more bandwidth. Our method can achieve a better tradeoff. It has the minimum energy consumption and wastes much less bandwidth than the ON-OFF scheme.

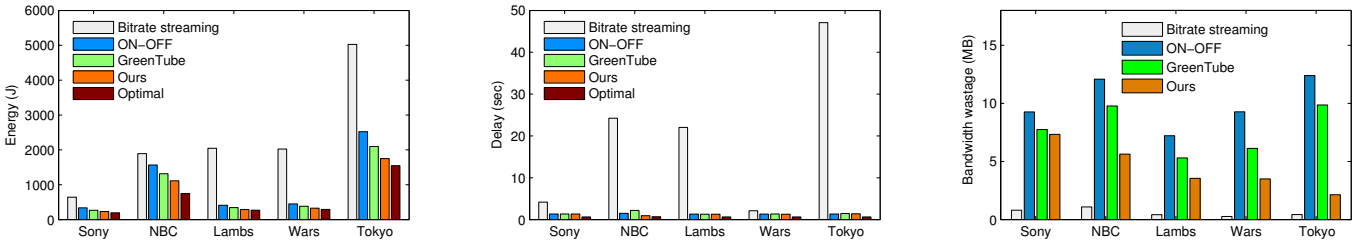
From the energy point of view, bitrate streaming consumes the most, since it always keeps the wireless interface on. ON-OFF and our method save 59.2% and 67.9% of energy than bitrate streaming, respectively. Our method saves 21.6% of energy than the ON-OFF method. For scenarios without skipping, all three methods consume less energy, since they only download part of the video within the skip duration. Bitrate streaming benefits the most as it downloads very little unused data. The energy reduction of our method is larger than ON-OFF since our method wastes much less bandwidth.

The delay of bitrate streaming is much higher than ON-



(a) Energy (b) Delay of bitrate streaming

Fig. 7. The energy and delay of different methods when users watch the whole video



(a) Energy (b) Delay (c) Bandwidth wastage

Fig. 8. The energy, delay and bandwidth wastage of different methods when users skip twice to watch a video

OFF and our method. Compared to the corresponding methods without skipping, bitrate streaming increases a little bit of delay, as its delay is mainly controlled by rebufferring. There are more delay increases in ON-OFF and our method since they need to download new GoPs if users skip out of the buffered data.

The ON-OFF method wastes more bandwidth, since it downloads new data even when there are still lots of data in the buffer. Our scheme reduces the bandwidth wastage by 59.4% when compared to ON-OFF, although it is still more than the bitrate streaming method. Considering both energy saving and delay reduction, our scheme is better.

C. Trace-driven Simulations

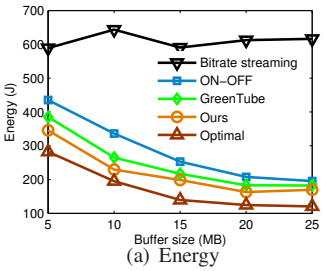
In the simulation, we use selected methods to download various videos with different length and data size. We also adjust parameters to evaluate their effects in different methods. The video traces are selected from [2], as listed in Table. II. All videos are using the H.264/AVC format with a definition of 352×288, and are compressed with a QP (quantization parameter) of 16. All videos have 30 frames per second. A GoP contains 16 frames and is formatted as IBBBPBBBPBBBPBBB, so each GoP has a constant duration of 0.53 second. The GoP is used as the minimum unit to download. We use LTE to stream the videos to smartphones. The energy model is described in Section III. For delay, we consider both initial delay and rebuffering delay.

1) Video Streaming without Skipping: First, we assume users do not skip and watch the whole video. Since the whole video is watched, no bandwidth is wasted. The energy consumption of different methods is shown in Fig. 7(a). Generally speaking, bitrate streaming consumes much more energy than other methods. The energy of the other four methods mainly depends on how long the wireless interface stays in the OFF state. ON-OFF saves some energy by turning wireless interface off. GreenTube works almost the same as ON-OFF as GreenTube tends to select the maximum buffer size. At the end of the video, GreenTube has some wrong predictions and uses small buffer size, so it consumes a little more energy. Our method outperforms ON-OFF and GreenTube because we turn on the wireless interface again when the buffered content is almost running out. The optimal method knows the next GoP size and thus can let the interface stay longer in the OFF state.

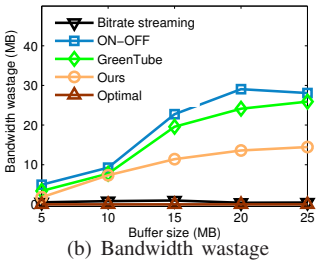
When comparing the power consumption of different videos, we notice that the energy of all methods, except bitrate streaming, is closely related to the data size of the video. For bitrate streaming, the energy is almost linearly related with the video length, since it keeps the wireless interface on almost all the time during video playback.

We then compare the delay of these five methods. Since there is no skipping, all methods except bitrate streaming only has an initial delay, which can be ignored. The delay of bitrate streaming is shown in Fig. 7(b). We notice that the delay is roughly related with video length, as the chance of rebuffering increases as time goes. One interesting thing is that video "Wars" has smaller delay than "Lambs", although they have the same length. The major reason is that the GoP size variation of "Wars" is smaller when compared to that of "Lambs", which can be seen from the difference between the maximum and minimum GoP size in Table II.

2) Video Streaming when Users Skip: In this test users skip twice to watch the videos, and the performance of various methods is shown in Fig. 8. The energy consumption is shown in Fig. 8(a), which has a similar trend as that without skipping. The difference is that every method saves some energy, since skipping reduces the amount of data to be downloaded. For delay comparison, bitrate streaming has much longer delay, which is consistent with that without skipping, and the optimal method only has one initial delay to download the first GoP. ON-OFF, GreenTube and our method have similar delay, since they all have one initial delay and two rebuffering delay, which

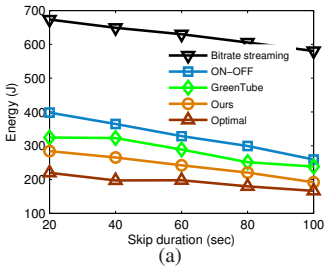


(a) Energy

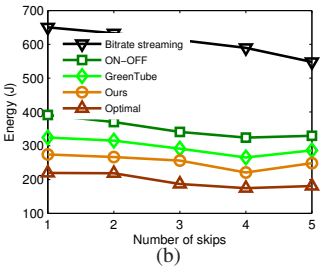


(b) Bandwidth wastage

Fig. 9. The impact of buffer size on energy and bandwidth wastage



(a)



(b)

Fig. 10. The impact of skip on energy

is caused by the two skips. For bandwidth wastage, GreenTube saves some bandwidth as it predicts the user watching time and selects a smaller buffer size when the user is about to skip. Our method works better than others since we use a small lower buffer threshold.

3) Impact of Parameters: We also evaluate the impact of parameters on energy consumption of various video streaming methods. As the performance of all methods is related with buffer size, we adjust the maximum buffer size from 5 to 25 MB and evaluate the energy consumption of various methods using the "Sony" video trace when users skip twice. For GreenTube, the candidate buffer size is adjusted proportionally. The result is shown in Fig. 9(a). The buffer size does not affect the energy of bitrate streaming, as it only uses a small buffer. For all other methods, the energy decreases when the buffer size increases, since downloading more data will leave the wireless interface in OFF state for a longer time. However, increasing the buffer size also increases bandwidth waste, as shown in Fig. 9(b). When the buffer size is larger than 10 MB, the energy saving rate slows down but the bandwidth wastage increases faster, so setting the buffer size to 10 MB can achieve a good balance.

We also evaluate the impact of user skipping on energy. We first adjust the skip duration by setting the skip number to 2 and then fix the skip duration to 60 seconds and adjust the skip number. The results are shown in Fig. 10(a) and Fig. 10(b), respectively. The results are similar in both cases. All methods consume less energy when the skip duration or skip number increases, since the total skipped video is increased, and energy can be saved by watching less video. In all cases, our method outperforms existing methods.

VIII. Conclusions

In this paper, we studied energy-aware video streaming on smartphones. There are two types of methods for video streaming: ON-OFF and bitrate streaming. ON-OFF can save energy but may waste bandwidth if users skip or quit the video. Bitrate streaming saves bandwidth wastage but introduces delay and consumes more energy. We first analyze the minimum energy to download a video in an offline manner. Then we introduce an online energy-aware solution to save energy, where video streaming is divided into stable and unstable modes, based on whether a user tends to watch video for a long time or tends to skip or quit. In the stable mode, we use an optimized ON-OFF scheme which reduces energy and bandwidth wastage. In the unstable mode, we use a modified bitrate streaming to reduce delay. We have implemented the online method on Android phones and evaluated the performance through testbed and trace-based simulations. Evaluation results show that our method can save large amount of energy and achieve a better tradeoff between delay and bandwidth compared to existing methods.

References

[1] Big buck bunny. http://www.bigbuckbunny.org/.
[2] H.264/avc video trace library. http://trace.eas.asu.edu/h264/index.html.
[3] Vlc media player. http://www.videolan.org/.
[4] Cisco visual networking index: Global mobile data traffic forecast update, 20132018. http://goo.gl/Atqj5d, 2014.
[5] D. H. Bui, K. Lee, S. Oh, I. Shin, H. Shin, H. Woo, and D. Ban. GreenBag: Energy-Efficient Bandwidth Aggregation for Real-Time Streaming in Heterogeneous Mobile Wireless Networks. In *IEEE Real-Time Systems Symposium (RTSS)*, 2013.
[6] P. M. Eittenberger, M. Hamatschek, M. Grossmann, and U. R. Krieger. Monitoring Mobile Video Delivery to Android Devices. In *ACM MMSys*, 2013.
[7] M. Hoque, M. Siekkinen, J. Nurminen, and M. Aalto. Dissecting mobile video services: An energy consumption perspective. In *IEEE WoWMoM*, 2013.
[8] M. A. Hoque, M. Siekkinen, and J. K. Nurminen. Using Crowd-sourced Viewing Statistics to Save Energy in Wireless Video Streaming. In *ACM MobiCom*, 2013.
[9] W. Hu and G. Cao. Energy Optimization Through Traffic Aggregation in Wireless Networks. In *IEEE INFOCOM*, 2014.
[10] W. Hu and G. Cao. Quality-Aware Traffic Offloading in Wireless Networks. In *ACM MobiHoc*, 2014.
[11] V. Joseph and G. de Veciana. NOVA: QoE-driven optimization of DASH-based video delivery in networks. In *IEEE INFOCOM*, 2014.
[12] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou. MicroCast: Cooperative Video Streaming on Smartphones. In *ACM MobiSys*, 2012.
[13] Krishnan, S. Shunmuga and Sitaraman, Ramesh K. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *ACM IMC*, 2012.
[14] X. Li, M. Dong, Z. Ma, and F. C. Fernandes. GreenTube: Power Optimization for Mobile Videostreaming via Dynamic Cache Management. In *Proceedings of the 20th ACM International Conference on Multimedia*, 2012.
[15] Y. Liu and M. Hefeeda. Video Streaming over Cooperative Wireless Networks. In *ACM MMSys*, 2010.
[16] R. Mittal, A. Kansal, and R. Chandra. Empowering Developers to Estimate App Energy Consumption. In *ACM MobiCom*, 2012.
[17] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network Characteristics of Video Streaming Traffic. In *ACM CoNEXT*, 2011.
[18] J. Yoon, H. Zhang, S. Banerjee, and S. Rangarajan. MuVi: A Multicast Video Delivery Scheme for 4G Cellular Networks. In *ACM MobiCom*, 2012.
[19] B. Zhao, Q. Zheng, and G. Cao. Energy-Aware Web Browsing in 3G Based Smartphones. In *IEEE ICDCS*, 2013.